Journal of Universal Computer Science, vol. 3, no. 9 (1997), 1074-1083
submitted: 14/8/97, accepted: 16/9/97, appeared: 28/9/97

Why We Need an Explicit Forum for Negative Results

—

Announcement of the Forum for Negative Results (FNR)

Lutz Prechelt
Universität Karlsruhe, Germany
prechelt@ira.uka.de

Abstract: Current Computer Science (CS) research is primarily focused on solving engineering problems. Often though, promising attempts for solving a particular problem fail for non-avoidable reasons. This is what I call a negative result: something that should have worked does not. Due to the current CS publication climate such negative results today are usually camouflaged as positive results by non-evaluating or mis-evaluating the research or by redefining the problem to fit the solution.
Such publication behavior hampers progress in CS by suppressing some valuable insights, producing spurious understanding, and misleading further research efforts. Specific examples given below illustrate and back up these claims.
This paper is the announcement of a (partial) remedy: a permanent publication forum explicitly for negative CS research results, called the *Forum for Negative Results, FNR*. FNR will be a regular part of J.UCS.

Key Words: FNR, forum, negative results, failures, research culture, progress.
Category: A.m, K.7.m, K.4.m, I.2.7 speech recognition, I.2.6 learning, C.1.3 data flow architectures, B.6.3 verification.

1 Current Computer Science research culture

Almost all Computer Scientists agree that Computer Science (CS) has a strong engineering problem-solving component: most of it ultimately aims at building useful systems and making them ever better and cheaper. This excludes only some parts of the theoretical CS research.

As a result, we tend to judge research contributions by their short or mid-term application usefulness: Those contributions are considered best that very successfully solve an important engineering problem. Contributions that solve only a small fraction of their problem are considered less good and contributions that entirely fail to solve their problem are often not considered research contributions at all. Consequently, most researchers strive for problem-solving successes, where the problems to be solved are selected from an engineering point of view.

On the other hand, most CS researchers also agree that CS needs a firm scientific base on which to build new engineering solutions. Such a base is provided when research emphasizes understanding more than engineering problem

solutions. In this case, obvious practical applicability needs not be a research quality attribute.

Note that the discrimination between problem-solving research and research for understanding is not a sharp one, it is only a matter of emphasis; see also the article of Fred Brooks in the March 1996 issue of CACM [Brooks 1996] and the responses in the July issue.

Looking at actual CS research contributions, a large majority of them makes claims that clearly belong to engineering problem-solving; a study conducted in 1994 found about 75 percent of all articles in a random CS article sample to be of this type [Tichy *et al.* 1995].

Thus, our emphasis in non-theoretical CS as a whole is quite clearly on engineering (which is not necessarily harmful) and our judgement of what constitutes good research is consequently biased (which is harmful).

The following sections will discuss how this bias slows down progress in CS and why a forum for negative results might help; I will give specific examples to illustrate my point. Then I will present the structure of the new *Forum for Negative Results* and finally discuss the most important objections against it, before concluding the paper with a call for comments.

2 Why and how it hurts CS

The CS research culture described above has negative impact on our discipline as a whole in three different ways: First, results that could contribute to understanding may be suppressed; second, results may be presented in misleading ways; and third, research may be misguided towards less fruitful directions. I will discuss these problems in order and intersperse specific examples for illustration. These examples are authentic reports of actual research; for sake of brevity and clarity of argument, their technical content was somewhat simplified.

2.1 Lost insights

If a failure is not published at all, obviously nobody else can learn from it. However, non-publication does not seem to be frequent in CS. More often, and more seriously, failures are published in a clouded, hidden form: Either the problem is redefined appropriately as to fit the solution obtained, or the "solution" is evaluated using only selected examples where it works well. Sometimes the solution is hardly evaluated at all. In all of these cases, the reasons for the failure are not analyzed and our understanding is improved less than it could have been.

2.2 Example 1: Estimating speaking rate

This example of non-publication I heard from members of a world-class speech recognition research group that wish to remain anonymous. It exemplifies the loss of an insight due to the complete suppression of a negative result.

Problem: Every speech recognition system expects a certain speaking rate. If a speaker speaks very fast, the error rate increases; the system only works well for roughly the speaking rate for which it was built — an important limitation that must be overcome. However, several separate subsystems can be built for

different speaking rates. Then it becomes necessary to estimate the speaking rate *before* the actual speech recognition in order to select the appropriate subsystem.

Idea: The basic layer of a speech recognition system is the set of phoneme models: There is one module for each of, say, 40 different phonemes and at each time each of these modules m estimates the probability that the current speech input represents m's phoneme. During a phoneme, the corresponding module is high and the others are low. During a transition from one phoneme to the next, all modules are low and therefore the entropy of the module signals is higher. The local maxima of the entropy time series should thus indicate the phoneme transitions, the frequency of which is a direct measure of speaking rate.

Result: The idea was implemented and tried with various sorts of phoneme models and for various speaking rates. Speaking rates were estimated for 2-second long blocks of speech. The correlation between estimated and actual speaking rate was between -0.15 (sic!) and 0.26, typically 0.12. The idea does not work at all; it is essentially useless.

Insight: The idea does not work because the output of the individual phoneme models is too unreliable; there are many spurious maxima in the entropy time series. This is true although the given phoneme models are among the best available world-wide. The accuracy of speech recognition systems mostly stems from the integration of other knowledge sources such as phoneme pair frequency distributions and dictionaries.

Handling: The idea, results, and analysis have not been and will not be published.

Effect: The maxima-counting idea is so appealing, even for somebody who knows about the unreliability of the individual phoneme models, that other speech recognition research groups will most probably steer into the same blind alley again once they find it.

2.3 Spurious insights

Spurious insights occur in one of three forms:

1. Sometimes dressed-up or distorted evaluations as mentioned above make the reader believe something that is not there. A solution may appear to possess properties that it does not have at all or only to a lesser degree.
2. Also, positive results sometimes occur by chance as shown in the example below. In both cases, such spurious results can seriously distort CS knowledge.
3. Most of the time, however, the authors only try to convey a success, but fail to convince the reader of it. As a result, a lot of CS research lacks credibility, which also harms CS as a whole.

The problem of cases 1 and 3 is lack of a fair and thorough evaluation of a solution. Several papers have published quantitative evidence or constructive critique in this matter, e.g. [Fenton *et al.* 1994; Prechelt 1996; Tichy *et al.* 1995]. One reason why we find such insufficient evaluations is that they would often produce negative results which would be hard to publish. The problem of case 2 is the non-publication of negative results as illustrated by the following example.

2.4 Example 2: Comparing neural network algorithms

The following is an example of *distributed research*. It shows how the suppression of negative results can lead to spurious positive results. The example is not

concrete, because it is impossible to find all the participants, but is quite realistic. It is discussed in similar form by Salzberg [Salzberg 1997].

Problem: Many researchers are trying to improve current neural network learning algorithms, e.g., for pattern classification tasks. Let us say that 20 different researchers are all trying to improve the same algorithm A.

Idea: Let us further assume that each of these 20 researchers has come up with the same modified version M of A and that this modification is useless: it neither improves nor reduces the performance of A. Both assumptions are realistic; there are many changes to learning algorithms that seem plausible as an improvement, but are neutral overall and that many people could come up with independently.

Result: Ideally, each of our 20 researchers will evaluate his or her change using multiple different problems and multiple training runs for each, and they will compare the results obtained for M to their results for A using a statistical significance test. Note that most neural network learning algorithms are indeterministic, because they start with a random initialization of the the network parameters. Therefore, the results of the significance test are indeterministic as well (after all, that is what statistical tests are for) and we can expect one of our 20 researchers to obtain significance at the 0.05 level.

Insight: The conclusion of this researcher will be that M is indeed better than A.

Handling: This one researcher will publish M as an improvement of A. Maybe one or two others also had (weakly) significant results and publish them as well. The other 17 will probably *not* publish their negative result that M is not a useful improvement of A although M seems appealing.

Effect: The research community will get the false impression that M is better than A — a spurious insight which is due to the suppression of the negative results obtained by the majority of researchers. The same could happen, although with smaller probability, even if M was in fact *worse* than A. The spurious positive result may further contribute to the evolution of false and misleading theories of learning that explain *why* M is supposed to be better than A.

2.5 Misdirected research

As a consequence of both lost or spurious insights, other researchers might pursue investigations that are ineffective or at least inefficient and that they would not have pursued, had the respective negative results been published with an appropriate analysis.

2.6 Example 3: Dataflow computers et al.

The history of dataflow computers is an example of misdirected research due to insufficient evaluation or insufficient consideration of negative *aspects* of research results. It bears many similarities to several other areas of CS; these could be discussed here just as well.

Problem: How to build efficient computer hardware given advancing electronics technology, beginning in the early 1970s.

Idea: Express a program as a dataflow graph and build hardware that can execute dataflow graphs. This will automatically exploit instruction-level parallelism by executing a program instruction when all of its input values are

available. Implicit control flow replaces the program counter. Programming becomes simpler and increasing numbers of transistors per chip can be utilized by increasing the number of basic execution units. Instruction-level parallelism is the most general kind of parallelism and thus will be more efficient than explicit parallelism with explicit synchronization [Dennis 1980].

Result: Operational prototypes of static dataflow computers were built in the mid-1970s. For most program domains, their parallelism was limited severely by the inability to execute loop iterations or recursive calls in parallel. This limitation is inherent in the principle of a static(!) dataflow computer. In the absence of sufficient instruction parallelism, sequential execution of dataflow computers is quite slow.

Insight: The insight at this point *should have been* that entirely replacing von Neumann style computing with dataflow computing might be the wrong idea. The next goal should have been understanding what *combination* of aspects of both paradigms would be most efficient [Iannucci 1990].

Handling: However, a decade after dataflow computers were first proposed and several years after the first prototypes became operational and the lack of instruction-level parallelism became apparent, one of the principal dataflow researchers still wrote: *"What are the prospects for data flow supercomputers? [...] A machine with up to 512 processing elements or cell blocks seems feasible."* [Dennis 1980]

Effect: Instead of working towards hybrid von-Neumann/dataflow architectures, the dataflow community went on to building dynamic dataflow machines [Hwang and Briggs 1984], which increased the amount of available parallelism but (predictably) suffered from its large token tagging, tag comparing, and data duplication overhead and was never able to keep up with von Neumann processors of similar cost.

A more critical evaluation and increased focus on problematic aspects of the dataflow idea could have saved significant amounts of research resources and had probably resulted in faster progress. As Hwang and Briggs diplomatically put it in 1984: *"Most advances are claimed by researchers in this area. The claimed advantages were only partially supported by performance analysis and simulation experiments. Operational statistics are not available from existing prototype data flow machines. Therefore, some of the claimed advantages are still subject to further verification."* [Hwang and Briggs 1984, p. 745]

CS research history features several other examples of similar nature, such as most areas of Artificial Intelligence (e.g. automatic program verification, pattern recognition, symbolic AI, neural network learning, fuzzy logic) computer integrated manufacturing (CIM), object-oriented methods, and others. In all of these cases, a lack of weakness analysis or the suppression of its negative results has led to avoidable waste of research resources and slowdown of progress, at least for a significant time during the early development phases of the respective areas.

Again: This is clearly not to say that all research in the above areas was wasted. Each of these areas has produced many valuable contributions to our knowledge. However, some *parts* of the work in these areas was wasted or inefficient and these parts were much larger than would have been necessary.

2.7 Example 4: Hardware verification

This final example illustrates how deeply researchers have absorbed the principle of not publishing negative results and how this can prevent useful analyses.

Problem: In formal hardware (VLSI) verification, several research groups have produced operational systems that all have different strengths and weaknesses. A comparative evaluation of these systems could be a powerful means of understanding *why* these weaknesses occur and might produce ideas for further improvements.

Idea: One researcher in this field, who wishes to remain anonymous, invited selected colleagues from all of these research groups to write a contribution to an edited monograph that was intended to provide such a comparative evaluation. Each group was given a set of example problems and was asked to elaborate which of these their system could handle well or not well and why. All of the papers were practically already accepted before they were even written, so there was no pressure towards producing only positive results — quite on the contrary: The editor asked explicitly to elaborate on weaknesses and their cause.

Result: Despite repeated explicit queries, several of the groups did hardly describe, let alone discuss, those cases where their system did not do well.

Effect: The comparative analysis was only half as useful as it could have been. Probably some important insights were lost.

3 A counter-measure: The *Forum for Negative Results*

Even if a meant-to-be problem-solving contribution fails and thus represents no direct engineering progress, it can be a useful research contribution: Quite often an analysis of the reasons why a particular approach to a problem failed could contribute to understanding, thus promoting further engineering advances and avoiding unfruitful research efforts.

The right thing to do with such a failure would thus be to publish a description and an analysis of the reasons instead of the now common disguised mis-presentation as a success. One major reason why such presentation of research as a negative result is so rare today is a lack of encouragement as discussed in Section 1.

This is why J.UCS hereby announces the establishment of a publication forum that *explicitly and exclusively* calls for negative results: The *Forum for Negative Results* (FNR). FNR is a permanent, specially marked section of J.UCS.

Call for Papers: The Forum for Negative Results (FNR)

The Forum for Negative Results (FNR) is a permanent special section of the Journal of Universal Computer Science (J.UCS) and exclusively publishes negative results, i.e., research that did not have the desired outcome, but still advances knowledge. J.UCS is an electronic journal published by Springer Verlag.

Rationale: As most of Computer Science is rather usefulness-oriented, it is currently difficult to publish work that demonstrates a non-progress, or negative result, with respect to usefulness. Therefore today,

- lessons to be learned from negative results are often lost, and
- many works tend to demonstrate neither progress nor non-progress.

FNR is a top-class forum for publishing Computer Science negative results that imply scientific insights. Just like J.UCS, FNR does not restrict contributions to particular topics. However, only papers with the following properties qualify for publication in FNR:

1. The work described had a clear goal, stated in the paper.
2. The starting point or approach of the work was promising and had plausible chances of success. These chances are explained in the paper.
3. Still, the goal was not met. The failure was not foreseeable during the implementation phase of the work; it was apparent only from the evaluation phase. (This rules out most purely theoretical work.)
4. There must be danger of somebody else trying a similar approach again, and failing.
5. Both implementation and evaluation were carried out according to highest scientific standards. These standards are documented in the paper.
6. At least part of the reason for the failure was understood in the evaluation or in subsequent analysis. The explanation is given in the paper. This explanation represents the scientific contribution of the paper.

FNR will be extremely selective. Any paper to appear in FNR must be *impeccable* with respect to points (1) and (5). High standards will also be applied to points (2), (3), and (4). As for point (6), the lesson learned must become clear, but no cure needs to be known.

Articles should be as concise as possible and concentrate on goals, approach, and reasons for failure instead of on technical details of implementation and evaluation. The reviewers of a paper submitted to FNR will apply these criteria when judging the contribution.

For further information see the FNR homepage at
http://wwwipd.ira.uka.de/fnr.

3.1 Additional remarks

Some points require particular emphasis: First, in order not to become a trash can of failed low-quality research, FNR will be very demanding. FNR will only

have a significant positive impact on the overall publication climate if it becomes a prestigious place for publishing one's research.

Second, whether an initial idea was really "promising" and the failure "unforeseeable" does obviously depend on the researchers' previous knowledge; others might have anticipated the problem. Here, similar reasonably high (but not too high) standards will be applied as for conventional J.UCS contributions.

Submissions to FNR will be reviewed and published like submissions to J.UCS, except that

- the FNR review criteria will be applied,
- the paper will appear with the subtitle "a contribution to the *Forum for Negative Results*", which also must be used by the author for submitting to FNR.

In particular, FNR contributions are in the same publication queue as J.UCS contributions. There is no fixed minimum or maximum number of FNR contributions in one J.UCS issue. No article submitted to FNR will ever directly compete with any non-FNR article submitted to J.UCS, as such competition could invalidate FNR's basic idea. Instead, when space becomes scarce, accepted articles will usually be published in order of submission. The typical publication delay, however, will be short compared to other high-class scientific journals.

4 Critique

Most people I talked to are in favor of the idea of having an explicit forum for negative results. Some others, though, disapprove. This section discusses their objections.

• **Science is not about truth, it is about status, money, and power.** This objection was posed by a few and seems to be considered correct by many — to some degree. However, all researchers I talked to also feel that science *should not* be this way. Moreover, the contradiction between truth and personal status that most people see connected with a failure is only a result of overemphasizing engineering success in Computer Science. Most researchers dislike the contradiction; so why not try and open new opportunities for removing it? FNR is such an opportunity.

• **Nobody likes failures.** (This one is closely connected to the one above.) Sure, no researcher likes admitting that something just did not work. However, most dislike disguise even more and will often prefer to analyze the failures they had as failures instead of feeling pressed to discuss or mis-evaluate them into successes.

• **Knowing the reasons for a failure is a competitive advantage that people will not give away.** For a few research areas that are both very focused and fast-moving, this may be true. In such an area, many research groups are targeting the same problems using similar methods and the area is leaping from success to success — and only success counts. In any really fast-moving area, on the other hand, the usual publication delay is sufficient for protecting the competitive advantage. Most areas of computer science are not critical in this respect, anyway.

• **A published failure may keep people from picking up the same idea later.** This fear is valid when a failure was only due to technological conditions

or constraints that might change later: If an idea was published as leading to a negative result, wouldn't other researchers be deterred from trying the idea again later? Wouldn't some successes be lost that way? No, they wouldn't — quite on the contrary: The contribution of the respective FNR paper would be to make the relevant technological constraints explicit and identify alternative conditions under which the same idea would probably work well. Such a description would foster, rather than discourage, picking up the idea again later, at exactly the right time.

• **High-quality negative results are also published elsewhere.** Right. Any paper FNR will accept would most probably also be accepted by other high-class journals. But that is not the point. The point is that today writing a paper in this way is daunting and getting it accepted is often unnerving. Hence, only a fraction of all failures that could be instructive is actually published and analyzed as negative results, thus damaging scientific progress. With FNR, more such useful negative results would appear.

As we see, on close examination all of this critique is invalid. Nevertheless, it may still adversely affect the success of FNR.

5 Conclusion

We cannot know in advance whether FNR will be accepted and will become a scientific success. However, if we want to improve the scientific quality of Computer Science research, we will have to give it a try.

Furthermore, J.UCS has an annotation facility whereby comments or additions can be attached to published articles. The annotation facility can be used by everyone: select "Discussion forum" (http://www.iicm.edu/jucs_annotations) on the JUCS homepage, register, and submit. All annotations are reviewed to ensure high quality. Discussions and additions bundled through the annotation mechanism will further improve the scientific impact of FNR contributions.

Hence, the Journal of Universal Computer Science proudly announces the *Forum for Negative Results* and encourages all authors to submit appropriate high-quality papers.

I also encourage every CS researcher to contribute an additional argument for or against FNR itself or an additional research war story, etc., by means of an annotation to the present article.

References

[Brooks 1996] Fred P. Brooks. The computer scientist as toolsmith II. *Communications of the ACM*, 39(3):61–68, March 1996.

[Dennis 1980] Jack B. Dennis. Data flow supercomputers. *IEEE Computer*, 13:48–56, November 1980.

[Fenton *et al.* 1994] Norman Fenton, Shari Lawrence Pfleeger, and Robert L. Glass. Science and substance: A challenge to the software engineering community. *IEEE Software*, 11(4):86–95, July 1994.

[Hwang and Briggs 1984] Kai Hwang and Fayé A. Briggs. *Computer Architecture and Parallel Processing.* McGraw-Hill, 1984.

[Iannucci 1990] Robert A. Iannucci. *Parallel Machines: Parallel Machine Languages: The Emergence of Hybrid Dataflow Computer Architectures.* Kluwer Academic Publishers, 1990.

[Prechelt 1996] Lutz Prechelt. A quantitative study of experimental evaluations of neural network learning algorithms: Current research practice. *Neural Networks*, 9(3):457–462, April 1996.

[Salzberg 1997] Steven L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3), 1997.

[Tichy *et al.* 1995] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28(1):9–18, January 1995. Also as TR 17/94 (August 1994), Fakultät für Informatik, Universität Karlsruhe, Germany, ftp.ira.uka.de.